Claim 20:

20. The apparatus of claim 19 wherein said monitor operates as executable code of the dispatcher algorithm used by an instruction processor when said instruction processor seeks a new task wherein a value related to a current level of busyness of said instruction processor using said monitor executable code is stored in an instruction processor busyness data area.

Claim 21:

21. The apparatus of claim 20 wherein said monitor evaluates the values stored in said instruction processor busyness data area and generates a level of busyness value of the multiprocessor system there-from, and based on a comparison between said current busyness value for this one instruction processor, produces a transfer affinity value, and wherein said monitor compares said transfer affinity value against a threshold level value to determine whether said another instruction processor will be permitted to steal a task from said one instruction processor-associated switching queue or not.

Claim 22:

22. A dispatcher algorithm for use in a multiple instruction processor computer system having at least three levels of memory, said three levels being at least two cache levels, a first of which is accessible directly by a single one of said instruction processors, a mid-level memory being a multiprocessor-accessible cache accessible by at least two of said instruction processors, and a third memory level being a main memory, accessible by all of said instruction processors, wherein tasks are directed to switching queues for cluster assignment on an affinity basis by an executive and said switching queues are maintained and controlled by said dispatcher algorithm, said dispatcher algorithm comprising:

an executable program for assigning affinity of each new task to a one of said clusters executing said executable program, wherein a cluster is a group of instruction processors,

a set of switching queues of substantially the same number as clusters wherein one switching queue is associated with said one of said clusters and a switching queue is also associated with substantially each other of said clusters, said switching queues having code for their operation and a data area wherein said data area is for maintaining a list of tasks for a cluster to accomplish,

a load balancing level matrix for directing said first cluster to steal a task from a switching queue associated with another cluster in accord with a predetermined mapping within said matrix when said first cluster is looking for an additional task.

Claim 23:

23. A dispatcher algorithm for use in a multiple instruction processor computer system having hierarchical levels of memory, wherein tasks are directed to switching queues for assignment to a processor unit on an affinity basis by an executive and said switching queues are maintained and controlled by said dispatcher algorithm, said dispatcher algorithm comprising:

an executable program for assigning affinity of each new task to a one of said processor units executing said executable program,

a set of switching queues of substantially the same number as processor units wherein one switching queue is associated with one of said processor units and a switching queue is also associated with substantially each other of said processor units, said switching queues having code for their operation and a data area wherein said data area is for maintaining a list of tasks for an processor unit to accomplish,

a load balancing level matrix for enabling said first processor unit to steal a task from a switching queue associated with another processor unit in accord

with a predetermined mapping within said matrix when said first processor unit is looking for an additional task.

Claim 24:

24. A multiple instruction processor computer system having a dispatcher algorithm as set forth in claim 23.

Claim 25:

25. A dispatcher algorithm as set forth in claim 23, wherein each said processor unit comprises either a cluster of instruction processors or a single instruction processor.

Claim 26:

26. A dispatcher algorithm as set forth in claim 25, wherein at least one of said processor units is comprised of a different number of instruction processors than at least one other one of said processor units.